



## 2.8 — Programs with multiple code files

👤 ALEX<sup>1</sup> 🕒 APRIL 20, 2024

### Adding files to your project

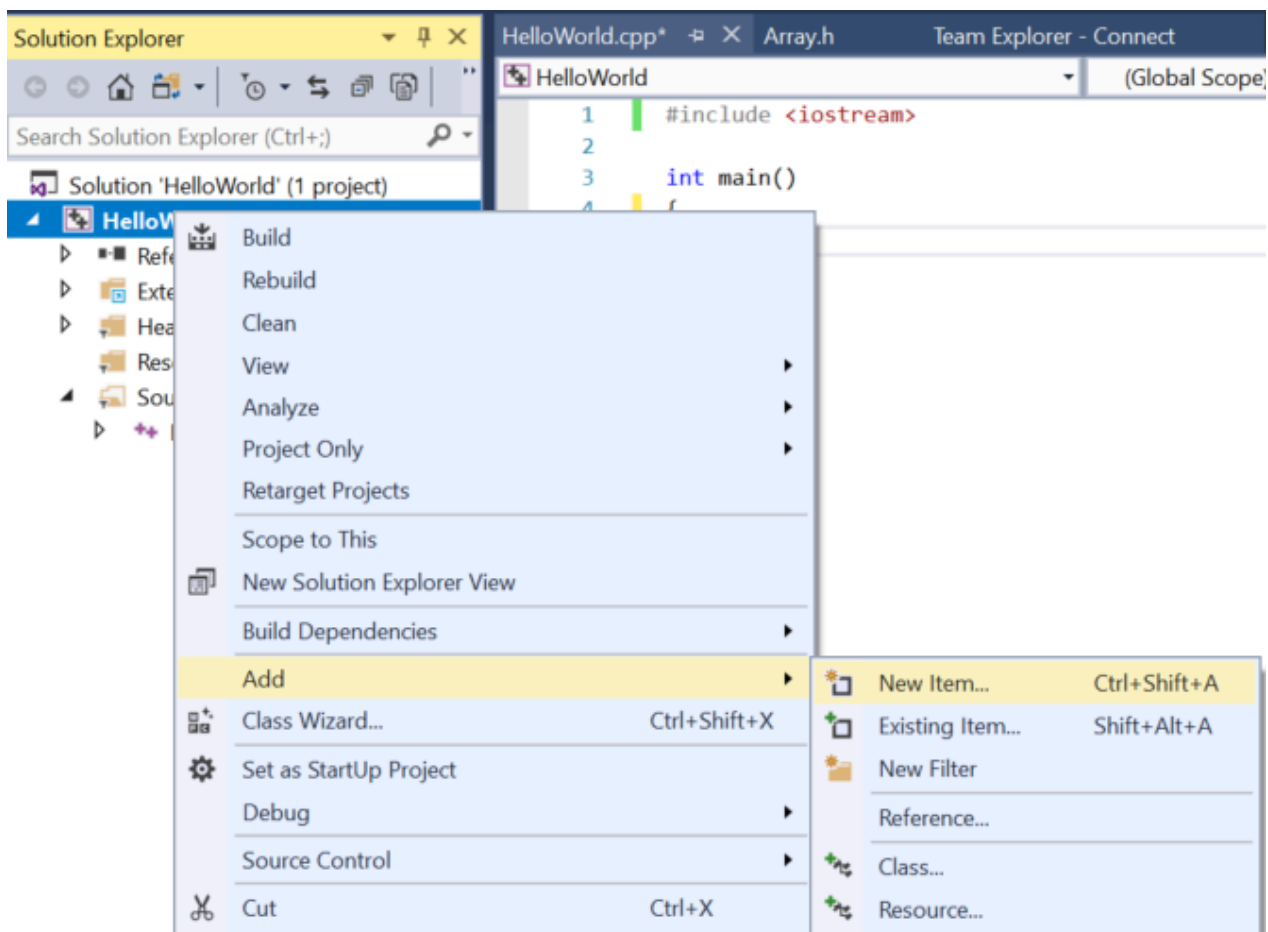
As programs get larger, it is common to split them into multiple files for organizational or reusability purposes. One advantage of working with an IDE is that they make working with multiple files much easier. You already know how to create and compile single-file projects. Adding new files to existing projects is very easy.

#### Best practice

When you add new code files to your project, give them a .cpp extension.

#### For Visual Studio users

In Visual Studio, right click on the *Source Files* folder (or the project name) in the Solution Explorer window, and choose *Add > New Item...*.



Make sure you have *C++ File (.cpp)* selected. Give the new file a name, and it will be added to your project.

Note: Your Visual Studio may opt to show you a compact view instead of the full view shown above. You can either use the compact view, or click “Show all Templates” to get to the full view.

Note: If you create a new file from the *File menu* instead of from your project in the Solution Explorer, the new file won't be added to your project automatically. You'll have to add it to the project manually. To do so, right click on *Source Files* in the *Solution Explorer*, choose *Add > Existing Item*, and then select your file.

Now when you compile your program, you should see the compiler list the name of your file as it compiles it.

## For Code::Blocks users

In Code::Blocks, go to the *File menu* and choose *New > File...*

In the *New from template* dialog, select *C/C++ source* and click *Go*.

You may or may not see a *welcome to the C/C++ source file wizard* dialog at this point. If you do, click *Next*.

On the next page of the wizard, select “C++” and click *Next*.

Now give the new file a name (don't forget the .cpp extension), and click the *All* button to ensure all build targets are selected. Finally, select *finish*.

Now when you compile your program, you should see the compiler list the name of your file as it compiles it.

## For GCC/G++ users

From the command line, you can create the additional file yourself, using your favorite editor, and give it a name. When you compile your program, you'll need to include all of the relevant code files on the compile line. For example: `g++ main.cpp add.cpp -o main`, where `main.cpp` and `add.cpp` are the names of your code files, and `main` is the name of the output file.

## For VS Code users

To create a new file, choose *View > Explorer* from the top nav to open the Explorer pane, and then click the *New File* icon to the right of the project name. Alternately, choose *File > New File* from the top nav. Then give your new file a name (don't forget the `.cpp` extension). If the file appears inside the `.vscode` folder, drag it up one level to the project folder.

Next open the `tasks.json` file, and find the line `"${file}", .`

You have two options here:

- If you wish to be explicit about what files get compiled, replace `"${file}",` with the name of each file you wish to compile, one per line, like this:

```
"main.cpp",  
"add.cpp",
```

- Reader “geo” reports that you can have VS Code automatically compile all `.cpp` files in the directory by replacing `"${file}",` with `"${fileDirname}\\*.cpp"` (on Windows).
- Reader “Orb” reports that `"${fileDirname}/*.cpp"` works on Unix.

## A multi-file example

In lesson [2.7 -- Forward declarations and definitions](https://www.learncpp.com/cpp-tutorial/forward-declarations/) (<https://www.learncpp.com/cpp-tutorial/forward-declarations/>)<sup>2</sup>, we took a look at a single-file program that wouldn't compile:

```
1  #include <iostream>
2
3  int main()
4  {
5      std::cout << "The sum of 3 and 4 is: " << add(3, 4) << '\n';
6      return 0;
7  }
8
9  int add(int x, int y)
10 {
11     return x + y;
12 }
```

When the compiler reaches the function call to `add` on line 5 of `main`, it doesn't know what `add` is, because we haven't defined `add` until line 9! Our solution to this was to either reorder the

functions (placing *add* first) or use a forward declaration for *add*.

Now let's take a look at a similar multi-file program:

*add.cpp*:

```
1 | int add(int x, int y)
2 | {
3 |     return x + y;
4 | }
```

*main.cpp*:

```
1 | #include <iostream>
2 |
3 | int main()
4 | {
5 |     std::cout << "The sum of 3 and 4 is: " << add(3, 4) << '\n'; // compile
6 |     error
7 |     return 0;
8 | }
```

Your compiler may compile either *add.cpp* or *main.cpp* first. Either way, *main.cpp* will fail to compile, giving the same compiler error as the previous example:

```
main.cpp(5) : error C3861: 'add': identifier not found
```

The reason is exactly the same as well: when the compiler reaches line 5 of *main.cpp*, it doesn't know what identifier *add* is.

Remember, the compiler compiles each file individually. It does not know about the contents of other code files, or remember anything it has seen from previously compiled code files. So even though the compiler may have seen the definition of function *add* previously (if it compiled *add.cpp* first), it doesn't remember.

This limited visibility and short memory is intentional, for a few reasons:

1. It allows the source files of a project to be compiled in any order.
2. When we change a source file, only that source file needs to be recompiled.
3. It reduces the possibility of naming conflicts between identifiers in different files.

We'll explore what happens when names do conflict in the next lesson ([2.9 -- Naming collisions and an introduction to namespaces](https://www.learncpp.com/cpp-tutorial/naming-collisions-and-an-introduction-to-namespaces/) (<https://www.learncpp.com/cpp-tutorial/naming-collisions-and-an-introduction-to-namespaces/>)<sup>3</sup>).

Our options for a solution here are the same as before: place the definition of function *add* before function *main*, or satisfy the compiler with a forward declaration. In this case, because function *add* is in another file, the reordering option isn't possible.

The solution here is to use a forward declaration:

main.cpp (with forward declaration):

```
1 | #include <iostream>
2 |
3 | int add(int x, int y); // needed so main.cpp knows that add() is a function
4 | defined elsewhere
5 |
6 | int main()
7 | {
8 |     std::cout << "The sum of 3 and 4 is: " << add(3, 4) << '\n';
9 |     return 0;
10 | }
```

add.cpp (stays the same):

```
1 | int add(int x, int y)
2 | {
3 |     return x + y;
4 | }
```

Now, when the compiler is compiling *main.cpp*, it will know what identifier *add* is and be satisfied. The linker will connect the function call to *add* in *main.cpp* to the definition of function *add* in *add.cpp*.

Using this method, we can give files access to functions that live in another file.

Try compiling *add.cpp* and the *main.cpp* with the forward declaration for yourself. If you get a linker error, make sure you've added *add.cpp* to your project or compilation line properly.

## Tip

Because the compiler compiles each code file individually (and then forgets what it has seen), each code file that uses `std::cout` or `std::cin` needs to `#include <iostream>`.

In the above example, if *add.cpp* had used `std::cout` or `std::cin`, it would have needed to `#include <iostream>`.

## Key insight

When an identifier is used in an expression, the identifier must be connected to its definition.

- If the compiler has seen neither a forward declaration nor a definition for the identifier in the file being compiled, it will error at the point where the identifier is used.
- Otherwise, if a definition exists in the same file, the compiler will connect the use of the identifier to its definition.
- Otherwise, if a definition exists in a different file (and is visible to the linker), the linker will connect the use of the identifier to its definition.

- Otherwise, the linker will issue an error indicating that it couldn't find a definition for the identifier.

## Something went wrong!

There are plenty of things that can go wrong the first time you try to work with multiple files. If you tried the above example and ran into an error, check the following:

1. If you get a compiler error about *add* not being defined in *main*, you probably forgot the forward declaration for function *add* in *main.cpp*.
2. If you get a linker error about *add* not being defined, e.g.

```
unresolved external symbol "int __cdecl add(int,int)" (?add@@YAHHH@Z) ref
```

2a. ...the most likely reason is that *add.cpp* is not added to your project correctly. When you compile, you should see the compiler list both *main.cpp* and *add.cpp*. If you only see *main.cpp*, then *add.cpp* definitely isn't getting compiled. If you're using Visual Studio or Code::Blocks, you should see *add.cpp* listed in the Solution Explorer/project pane on the left or right side of the IDE. If you don't, right click on your project, and add the file, then try compiling again. If you're compiling on the command line, don't forget to include both *main.cpp* and *add.cpp* in your compile command.

2b. ...it's possible that you added *add.cpp* to the wrong project.

2c. ...it's possible that the file is set to not compile or link. Check the file properties and ensure the file is configured to be compiled/linked. In Code::Blocks, compile and link are separate checkboxes that should be checked. In Visual Studio, there's an "exclude from build" option that should be set to "no" or left blank.

3. Do *not* `#include "add.cpp"` from *main.cpp*. This will cause the preprocessor to insert the contents of *add.cpp* directly into *main.cpp* instead of treating them as separate files.

## Summary

C++ is designed so that each source file can be compiled independently, with no knowledge of what is in other files. Therefore, the order in which files are actually compiled should not be relevant.

We will begin working with multiple files a lot once we get into object-oriented programming, so now's as good a time as any to make sure you understand how to add and compile multiple file projects.

Reminder: Whenever you create a new code (.cpp) file, you will need to add it to your project so that it gets compiled.

## Quiz time

### Question #1



Split the following program into two files (main.cpp, and input.cpp). main.cpp should have the main function, and input.cpp should have the getInteger function.

[\(javascript:void\(0\)\)](#)<sup>4</sup>

Hint: Don't forget that you'll need a forward declaration in main.cpp for function getInteger().

```
1 | #include <iostream>
2 |
3 | int getInteger()
4 | {
5 |     std::cout << "Enter an integer: ";
6 |     int x{};
7 |     std::cin >> x;
8 |     return x;
9 | }
10 |
11 | int main()
12 | {
13 |     int x{ getInteger() };
14 |     int y{ getInteger() };
15 |
16 |     std::cout << x << " + " << y << " is " << x + y << '\n';
17 |     return 0;
18 | }
```

[Hide Solution](#) [\(javascript:void\(0\)\)](#)<sup>4</sup>

input.cpp:

```
1 | #include <iostream> // we need iostream since we use it in this file
2 |
3 | int getInteger()
4 | {
5 |     std::cout << "Enter an integer: ";
6 |     int x{};
7 |     std::cin >> x;
8 |     return x;
9 | }
```

main.cpp:

```
1 | #include <iostream> // we need iostream here too since we use it in this
2 |   file as well
3 |
4 | int getInteger(); // forward declaration for function getInteger
5 |
6 | int main()
7 | {
8 |     int x{ getInteger() };
9 |     int y{ getInteger() };
10 |
11 |     std::cout << x << " + " << y << " is " << x + y << '\n';
12 |     return 0;
13 | }
```



## [Next lesson](#)

2.9 [Naming collisions and an introduction to namespaces](#)

3



## [Back to table of contents](#)

5



## [Previous lesson](#)

2.7 [Forward declarations and definitions](#)


2

6



**B** **U** **URL** **INLINE CODE** **C++ CODE BLOCK** **HELP!**

Leave a comment...

 Name\*

@ Email\* | ?

Notify me about replies:



POST COMMENT

🔧 Find a mistake? Leave a comment above!?

👤 Avatars from <https://gravatar.com/><sup>8</sup> are connected to your provided email address.

690 COMMENTS

Newest ▼



derek

🕒 April 21, 2024 2:01 pm

If you can't see the add existing files or things aren't working as expected. You can hit the "Switch between solutions and available views" and double click your .sln file.

I was stuck in the folder view and confused as to why I couldn't add existing code files.



0



Reply

**Anon**

🕒 March 21, 2024 8:18 am

When I compile main.cpp with the add(int x, int y) forward declaration, it works and all, but I can't see add.cpp in the output:

Build started at 11:10 AM...

```
1>----- Build started: Project: HelloWorld, Configuration: Debug x64 ---
---
1>main.cpp
1>HelloWorld.vcxproj →
C:\Users\computer\source\repos\HelloWorld\x64\Debug\HelloWorld.exe
===== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped
=====
===== Build completed at 11:10 AM and took 01.184 seconds =====
```

It doesn't really matter for now, as the program is trivial. But in larger programs with multiple files, how can I see that every file was compiled properly? AKA How can I enable a way to see all the files in the output?



0



Reply

**Alex**

Author

🗨 Reply to [Anon](#)<sup>9</sup> 🕒 March 21, 2024 2:48 pm

This shouldn't compile (unless you had your main.cpp #include add.cpp, which is bad). You need to add add.cpp to your project, so that it will show up in the list of compiled files.



0



Reply

**Ryan Rawlings**

🕒 March 9, 2024 10:26 am

VS Code has shorthand for path separators so you don't have to define tasks differently on windows and linux:

`${/}` - shorthand for `${pathSeparator}`

[https://code.visualstudio.com/docs/editor/variables-reference#\\_predefined-variables](https://code.visualstudio.com/docs/editor/variables-reference#_predefined-variables)



0



Reply

**Jordan**

🕒 March 4, 2024 6:00 pm

I am a little confused with this chapter and my question is; won't it be necessary to link the different files together using header files link like `#ifndef` followed by the name\_h, `#define` followed by the name\_h in the main cpp file?

How do we determine the necessity of a different file if we can declare our functions in the main cpp?

Also, I am using Replit as my IDE. Would you recommend using VS instead?

👍 0

➡ Reply

**Alex**

Author

👤 Reply to [Jordan](#)<sup>10</sup> 🕒 March 6, 2024 9:34 pm

Header files aren't linked, they are included. We discuss header files (and header guards, which is what the `#ifndef` stuff is) later in this chapter.

I don't have any familiarity with Replit, but VS is free. You can have more than one compiler installed. Try it and see if you like it.

👍 1

➡ Reply

**jonisu2**

🕒 February 28, 2024 7:37 am

In the second photo of this page of tutorial in "Installed" section I can't find the "Visual C++" section, and I don't understand where to get it from. Please help.

👍 0

➡ Reply

**Alex**

Author

👤 Reply to [jonisu2](#)<sup>11</sup> 🕒 March 2, 2024 2:18 pm

Newer versions of Visual Studio have a compact view that minimizes a lot of the interface. If you click "Show all Templates" you will see the full view.

👍 0

➡ Reply

**jonisu2**👤 Reply to [Alex](#)<sup>12</sup> 🕒 March 10, 2024 10:11 am

But where to find that "Show all Templates"? I can't seem to find it anywhere.

👍 0

➡ Reply

**Alex** AuthorReply to [jonisu2](#)<sup>13</sup> ⌚ March 11, 2024 5:37 pm

In the bottom left corner of the compact interface. What are you actually seeing?

👍 0 ➡ Reply

**jonisu2**Reply to [Alex](#)<sup>14</sup> ⌚ March 13, 2024 10:38 am

Oh, I was on Property Manager. My bad! In the Solution Explorer, I instantly found the "Add new item". Thanks!  
(but seriously, sorry for being so foolish .\_.)

👍 0 ➡ Reply

**AVK**

⌚ February 22, 2024 8:53 pm

**It reduces the possibility of naming conflicts between identifiers in different files.**

As I understand this:

This is related to the concept of forward declarations. Having to compile one file with no memory of compilation of other files means that, we can only concern ourselves with the forward declarations in the particular file that we are working on. If all the files were to be compiled at the same time, chances of a declaration in one file is colliding with another is increased significantly. This kind of collision can be detected by the linker later on.

Is this understanding correct?

📝 Last edited 2 months ago by AVK

👍 0 ➡ Reply

**Alex** AuthorReply to [AVK](#)<sup>15</sup> ⌚ February 23, 2024 5:23 pm

Yes, and a collision between a forward declaration and some other identifier in a file can often be caught by the compiler, which is faster than waiting until the linker runs.

👍 0 ➡ Reply

**IceFloe**

⌚ February 20, 2024 6:25 pm

To create a C++ project with two files `main.cpp` and `input.cpp` in the `nvim` editor on Termux, I did this:

1. Created a new project (in turn):

```
1 | mkdir myproject
2 | cd myproject
```

2. Initialized the git repository:

```
1 | git init
```

3. Created files using `nvim`:

```
1 | nvim main.cpp
2 | nvim input.cpp
```

4. Wrote the code of these files according to the task.

5. Created a Makefile:

```
1 | nvim Makefile
```

6. Added compilation rules there (`main.o` `input.o` object files):

```
1 | main: main.o input.o
2 |     g++ main.o input.o -o main
3 |
4 | main.o: main.cpp
5 |     g++ -c main.cpp
6 |
7 | add.o: input.cpp
8 |     g++ -c input.cpp
```

7. Compiled:

```
1 | make
```

Now I have a ready-made project (which works) with two source code files, which I launched using `./main`.

I searched for additional information on the Internet, but even despite the difficulties, I was wondering exactly how it all works, I'm sure it was possible to find simpler options, but I'm still learning and I want to take into account all the nuances

 Last edited 2 months ago by IceFloe

 1  Reply

**Ian**

🕒 February 17, 2024 12:26 pm

"\${fileDirname}" works on mac.

👍 0

➡ Reply

**Howard**

🕒 February 16, 2024 5:23 am

For fellow Mac vs code users:

I tried several ways to compile multiple files in vs code, but they did not work. If the same happens to your vs code, try to type `sudo ./your file name` in your command line. This works well on my vs code, and I hope this could help :)

👍 0

➡ Reply

**Rishi**

🕒 February 11, 2024 9:32 am

I am not able to figure out how to compile multiple cpp files in the same folder in VScode (FOR MAC).

Please help!!

👍 0

➡ Reply

**asfadfasdfsafasdf**🗨 Reply to [Rishi](#)<sup>16</sup> 🕒 February 12, 2024 9:17 am

`g++ -o sum main.cpp add.cpp` - worked for me

👍 0

➡ Reply

**ask**

🕒 February 8, 2024 11:41 am

For the ppl like me that prefers to compile at the terminal you can do something like this on your terminal

```
g++ -o main *.cpp
```

👍 1

➡ Reply

**Dongbin**

🕒 January 28, 2024 1:29 pm

"It reduces the possibility of naming conflicts between identifiers in different files." <-- Why does requiring forward declaration in individual files accomplish this? Any help is appreciated.

👍 0

➡ Reply

**Alex**

Author

➡ Reply to [Dongbin](#)<sup>17</sup> 🕒 January 31, 2024 7:30 pm

Forward declarations provide a way for us to selectively declare names we care about. We only have to worry about whether the names we forward declare conflict with something else in our file. If all names could be accessed in any file, then any name could conflict with any name, which would significantly increase the chance of collision.

👍 0

➡ Reply

**Skai**

🕒 January 27, 2024 9:50 am



Forgot the parenthesis like a pro B)

👍 1

➡ Reply

**Swaminathan**

🕒 January 22, 2024 5:25 am

Hello Alex, you say `#include "add.cpp"` is bad which is clear. But how is it different and efficient from `#include <iostream>` which you say is mandatory?

🔗 *Last edited 3 months ago by Swaminathan*

👍 0

➡ Reply

**Swaminathan**➡ Reply to [Swaminathan](#)<sup>18</sup> 🕒 January 23, 2024 1:03 am

Please never mind: I found sufficient explanation in the upcoming lesson <https://www.learncpp.com/cpp-tutorial/header-files/>

👍 0

➡ Reply

**shivam**

🕒 January 19, 2024 5:55 am



help me out with vs code multi program file linking together ,i don't understand what you did in the vs code box.

👍 0    ➡ Reply



**robert**

🔗 Reply to [shivam](#)<sup>19</sup>    ⌚ January 24, 2024 1:01 am

what the author is doing is basically create another .cpp file in the same folder as your main code(the one with int main). if you really don't understand you can just make new file, which i think you already know, and save it to the same directory folder.

👍 0    ➡ Reply



**Violet**

⌚ December 7, 2023 2:54 am

For VS Code block on how to link files together, it is written to replace `"${file}"`, with `"${fileDirname}\\*.cpp"` but it should be, replace `"${file}"`, with `"${fileDirname}\\*.cpp"`, (there is no comma in original)

👍 3    ➡ Reply



**ArminC**

⌚ December 4, 2023 4:14 am

Configuring **Visual Studio Code** for a project with multiple source files -

Running Code (Code Runner extension):

- Go to **Extension Settings**, find **Code-runner: Executor Map**, press **Edit in settings.json** and at line **.cpp** modify it as follows: `"cpp": "cd $dir && g++ -std=c++23 *.cpp -o $fileNameWithoutExt && $dir$fileNameWithoutExt"`, where `$fileName` was changed to `*.cpp`.

Running C/C++ File:

- Go to **tasks.json** and under args, replace `"${file}"`, with `"${fileDirname}\\*.cpp"`, .

The above setup is made on Windows. For Unix systems try to replace `\\` with `/`.

🔗 Last edited 4 months ago by ArminC

👍 9    ➡ Reply

**Ron**Reply to [ArminC](#)<sup>20</sup> ⌚ December 26, 2023 4:54 pm

Thank you so much, this saved me hours of pulling my hair out. I fixed the task.json but not the coderunner one.

👍 2 ➡ Reply

**tatadott**

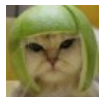
⌚ November 11, 2023 10:05 am

"If a definition exists in the same file, the compiler will connect the use of the identifier to its definition."

Does the above stand true if there are two definitions of the same function one in main.cpp and other in add.cpp?

🔗 Last edited 5 months ago by tatadott

👍 0 ➡ Reply

**Alex** AuthorReply to [tatadott](#)<sup>21</sup> ⌚ November 11, 2023 2:21 pm

Assuming both definitions are visible to the linker, the linker will error, as this is an ODR violation.

If either or both of the definitions are not visible to the linker, there is no issue.

👍 1 ➡ Reply

**thisisfun**

⌚ November 8, 2023 10:31 am

Note: You can also do `#include <FILENAME>` Example: `#include </home/HOME/YOURFOLDER/YOURC++CODE.cpp>` on linux!

👍 0 ➡ Reply

**thisisfun**Reply to [thisisfun](#)<sup>22</sup> ⌚ November 8, 2023 10:34 am

I just read More and found out you cant do `#include <Filename>`  
Well that sucks so dont use that

👍 0 ➡ Reply

**HimanshuK**

🕒 October 23, 2023 12:36 pm

I am using code runner extension in VS code, my code runs fine but i can see it only generates 1 .exe file ie: main.exe. Is it fine if its not making a add.exe file? All my code is same as the once in example.

👍 0

➡ Reply

**L1NTHALO**🗨 Reply to [HimanshuK](#)<sup>23</sup> 🕒 November 3, 2023 1:27 pm

If I understood correctly then that should be normal. Main.cpp and add.cpp get compiled then linked and then turned into only one executable file.

👍 0

➡ Reply

**Anything**

🕒 October 13, 2023 1:51 am

Lets, say we have 2 files in a folder we want to link and run .If we have compiled both of the files seperately, and then try to run the file with the main function. While it run? Or do we need 2 files even though they may have been compiled beforehand seperately to compile together at the same time for them to be linked

👍 1

➡ Reply

**Alex** Author🗨 Reply to [Anything](#)<sup>24</sup> 🕒 October 14, 2023 1:32 pm

You'll need to link both files into an executable first. If only one of the files has a main() function defined, this should work. If both do, the linker should give you a redeclaration error.

👍 1

➡ Reply

**handle-sp**

🕒 October 8, 2023 10:25 am

XCode (v.15.0) is failing to compile the add.cpp file because there is no function prototype (forward declaration) for the add() function in the add.cpp file, even though I was adding a forward declaration of the add() function in main.cpp.

For anyone using XCode (v.15.0) on MacOS, I turned off the following build setting for both Debug and Release:

- Missing Function Prototypes => NO

After disabling this build setting I was able to compile/link/execute successfully the example code.

👍 0    ➡ Reply

**idk**

🕒 October 4, 2023 11:23 am

tried doing this using the code you provided(main.cpp, add.cpp)(they are in the same directory) and when I run the main.cpp I get this error:

```
C:/msys64/mingw64/bin/./lib/gcc/x86_64-w64-mingw32/13.2.0/./.././../x86_64-w64-mingw32/bin/ld.exe: C:\Users\10\AppData\Local\Temp\ccRYNny3.o:main.cpp:(.text+0x37)
undefined reference to `add(int, int)'
collect2.exe: error: ld returned 1 exit status
help?
```

👍 0    ➡ Reply

**Alex**    Author

👤 Reply to [idk](#) <sup>25</sup>    🕒 October 5, 2023 1:46 pm

Your linker isn't connecting the call to add() to the definition of add(). This likely means you forgot to add add.cpp to your project, so it isn't being compiled/linked in.

👍 0    ➡ Reply

**idk**

👤 Reply to [Alex](#) <sup>26</sup>    🕒 October 5, 2023 2:15 pm

I compiled the program using clang  
it gives me:

- Executing task: & 'C:/Program Files/LLVM/bin/clang++' -std=c++20 -fdiagnostics-color=always -g main.cpp add.cpp -o D:\something\something\C++\main.exe

LINK : D:\something\something\C++\main.exe not found or not built by the last incremental link; performing full link

- Terminal will be reused by tasks, press any key to close it.

my tasks.json file is:({

// See <https://go.microsoft.com/fwlink/?LinkId=733558>

// for the documentation about the tasks.json format

"version": "2.0.0",

"tasks": [

{

"type": "shell",

```

"label": "C/C++: clang++.exe build active file",
"command": "C:/Program Files/LLVM/bin/clang++",
"args": [
  "-std=c++20",
  "-fdiagnostics-color=always",
  "-g",
  "main.cpp",
  "add.cpp",
  "-o",
  "${fileDirname}\\${fileBasenameNoExtension}.exe"
],
"options": {
  "cwd": "${workspaceFolder}"
},
"group": {
  "kind": "build",
  "isDefault": true
},
"detail": "Task generated by Debugger.",
"problemMatcher": []
}
]
}
)

```

and it creates a main.exe file but when I run main in vs codium(using the main.exe file in the directory does nothing) it just give me this(same problem) :

```

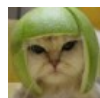
C:/msys64/mingw64/bin/./lib/gcc/x86_64-w64-mingw32/13.2.0/././././x86_64-w64-
mingw32/bin/ld.exe: C:\Users\10\AppData\Local\Temp\ccNUNMZA.o:main.cpp:
(.text+0x37): undefined reference to `add(int, int)'
collect2.exe: error: ld returned 1 exit status

```

error and deleted the main.exe file

 Last edited 6 months ago by idk

 0  Reply



**Alex** Author

 Reply to [idk](#) <sup>27</sup>  October 5, 2023 9:41 pm

Not sure. Sounds like something in your VS Code configuration, but I don't run VS Code so I'm not sure what.

 0  Reply



**idk**

🗨️ Reply to [Alex](#)<sup>28</sup> 🕒 October 8, 2023 9:03 am

maybe this can help

tried to build using cmake, which cmake worked but when I build the project using ms I got this

```
C:\Program Files\Microsoft Visual
Studio\2022\Community\VC\Tools\MSVC\14.37.32822\bin\HostX64\x86\
CL.exe /c /Zi /nol
go /W3 /WX- /diagnostics:column /Od /Ob0 /Oy- /D _MBCS /D WIN32 /D
_WINDOWS /D "CMAKE_INTDIR=\"Debug\"" /Gm- /EHsc /R
TC1 /MDd /GS /fp:precise /Zc:wchar_t /Zc:forScope /Zc:inline /GR
/Fo"main.dir\Debug\" /Fd"main.dir\Debug\vc143.pdb"
/external:W3 /Gd /TP /analyze- /errorReport:queue
D:\something\something\C++\buildhere\main.cpp
D:\something\something\C++\buildhere\add.cpp
main.cpp
add.cpp
Generating Code...
Link:
C:\Program Files\Microsoft Visual
Studio\2022\Community\VC\Tools\MSVC\14.37.32822\bin\HostX64\x86\li
nk.exe /ERRORREPO
RT:QUEUE
/OUT:"D:\something\something\C++\buildhere\bla\Debug\main.exe"
/INCREMENTAL /ILK:"main.dir\Debug\main.ilk" /NOLOG
O kernel32.lib user32.lib gdi32.lib winspool.lib shell32.lib ole32.lib
oleaut32.lib uuid.lib comdlg32.lib advapi32.li
b /MANIFEST /MANIFESTUAC:"level='asInvoker' uiAccess='false'"
/manifest:embed /DEBUG /PDB:"D:\something\something\C++\buil
dhere\bla\Debug\main.pdb" /SUBSYSTEM:WINDOWS /TLBID:1
/DYNAMICBASE /NXCOMPAT
/IMPLIB:"D:/something/something/C++/buildhere
/bla/Debug/main.lib" /MACHINE:X86 /SAFESEH /machine:X86
main.dir\Debug\main.obj
main.dir\Debug\add.obj
MSVCRTD.lib(exe_winmain.obj) : error LNK2019: unresolved external
symbol _WinMain@16 referenced in function "int __cdecl
I invoke_main(void)" (?invoke_main@@@YAHXZ)
[D:\something\something\C++\buildhere\bla\main.vcxproj]
D:\something\something\C++\buildhere\bla\Debug\main.exe : fatal error
LNK1120: 1 unresolved externals [D:\something\something\C++
\buildhere\bla\main.vcxproj]
Done Building Project
"D:\something\something\C++\buildhere\bla\main.vcxproj" (default
targets) -- FAILED.
```

Done Building Project

"D:\something\something\C++\buildhere\bla\main.vcxproj.metaproj"  
(default targets) -- FAILED.

Done Building Project

"D:\something\something\C++\buildhere\bla\main.sln" (default targets) -  
- FAILED.

Build FAILED.

"D:\something\something\C++\buildhere\bla\main.sln" (default target)  
(1) ->

"D:\something\something\C++\buildhere\bla\main.vcxproj.metaproj"  
(default target) (3) ->

"D:\something\something\C++\buildhere\bla\main.vcxproj" (default  
target) (4) ->

(Link target) ->

MSVCRTD.lib(exe\_winmain.obj) : error LNK2019: unresolved external  
symbol \_WinMain@16 referenced in function "int \_\_cd  
ecl invoke\_main(void)" (?invoke\_main@@YAHXZ)

[D:\something\something\C++\buildhere\bla\main.vcxproj]

D:\something\something\C++\buildhere\bla\Debug\main.exe : fatal error  
LNK1120: 1 unresolved externals

[D:\something\something\C++\buildhere\bla\main.vcxproj]

0 Warning(s)

2 Error(s)

👍 0

➡ Reply



**Alex** Author

🗨 Reply to [idk](#)<sup>29</sup> 🕒 October 10, 2023 1:48 pm

> unresolved external symbol \_WinMain@16

This means you set your project to build a windows application,  
not a console application. Change it to a console application and  
try again.

👍 0

➡ Reply



**Yopi**

🕒 September 30, 2023 3:51 am

Thanks for the lesson

👍 1

➡ Reply

**Gamborg**

🕒 September 15, 2023 8:33 am

**Question: Is there a reason to put the forward declarations outside the main body. Or can you place the forward declaration on the line before the first time you use a function from a different file?**

Such a nice and clean main(). I would show what's happening, but its all spread out into 16 different files now. To my surprise and skepticism, everything worked at first compile attempt.

```
1  #include <iostream>
2
3  int main()
4  {
5
6      void divideTwoNumbersTogether();
7      divideTwoNumbersTogether();
8
9      void doubleInteger();
10     doubleInteger();
11
12     void doubleAndTripleInteger();
13     doubleAndTripleInteger();
14
15     void checkEquality();
16     checkEquality();
17
18     void anotherRandomIntegerHandling();
19     anotherRandomIntegerHandling();
20
21     return 0;
22 }
```

👍 0

➡ Reply

**anon**🗨 Reply to [Gamborg](#)<sup>30</sup> 🕒 September 16, 2023 11:49 pm

Better to declare functions in the global scope and not inside main(). This way the declared function is accessible to all other functions and not just main().

👍 2

➡ Reply

**Gamborg**🗨 Reply to [anon](#)<sup>31</sup> 🕒 September 17, 2023 12:12 am

I can see that global scope is something covered in chapter 6 apparently. Thank you for the help, I will move them outside main().

👍 0

➡ Reply



**allow anonymous commenting**

🕒 September 12, 2023 2:13 pm

for future rewrite rather than loosely mentioning it in a comment at the very end in an exercise example you probably should explain more discretely why `#include<iostream>` is necessary in both files as that could probably be confusing for beignners

👍 1    ➡ Reply

**Cpp Learner**

🕒 September 12, 2023 5:18 am

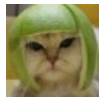
If I understand correctly it is bad practice to use `#include "file.cpp"` to include source files into another file and that `#include <filename>` should be reserved for header files only (i.e. `#include "file.hpp"`)

So a header just adds a forward declaration into the cpp file via preprocessor directive and each source file should and can be compiled independently from one another that way.

Is this correct?

I'm guessing you probably cover this in detail in later sections, but for now I thought I'd drop this comment in case anyone else is wondering the same thing

👍 0    ➡ Reply

**Alex** Author🗨 Reply to Cpp Learner<sup>32</sup> 🕒 September 12, 2023 12:29 pm

Yep!

👍 1    ➡ Reply

**Cpp Learner**🗨 Reply to Cpp Learner<sup>32</sup> 🕒 September 12, 2023 6:14 am

You did :)

Just after this section - [https://www.learncpp.com/cpp-tutorial/header-files/#corresponding\\_include](https://www.learncpp.com/cpp-tutorial/header-files/#corresponding_include)

👍 0    ➡ Reply

**Huu**

🕒 September 8, 2023 6:14 pm

You have discussed about "\${fileDirname}/\*\*/\*.cpp", where it will compile all files with extension .cpp at the end. However, I do not know the cmd line for terminal to compile all cpp.

 Last edited 7 months ago by Huu

 0  Reply



**Rohak**

 August 22, 2023 11:36 am

For MacOS VS Code, I found some difficulty in compiling multiple files at once, especially in the quiz question, and I expect some other users may find this difficulty as well.

Here's my comment from StackOverflow regarding this:

"Cool I figured how to compile all the files in my current folder.

1. Update the Clang version in tasks.json by adding a "-std=c++20", in my args
2. Replace "\${file}" with "\*.cpp" to account for all the files compiling in my build (replace star with individual files in each line for specific files). "

 0  Reply



**Rishi**

 Reply to [Rohak](#)<sup>33</sup>  February 10, 2024 8:37 pm

still showing error : linker command failed .  
please help!!

 0  Reply



**Neil**

 August 7, 2023 9:44 pm

I got a bit confused about the forward declaration, because the declaration and the add.cpp file had the same name. From that I thought the additional file name had to have its own declaration. Eventually looking at the solution resolved that.

Aside from that, my solution was very close to the example. It would compile, but when run, it would display the text "Enter an integer: " and not respond to any numeric entry. It was essentially locked, and I had to close the window to exit it.

To diagnose, I included a std::cout line to display the number being entered. Then after rebuilding, it would work, but it displayed the number twice on two different lines.

Here's my code:

```
// input.cpp; input module for exercise 2.8 question1
```

```
#include <iostream>

int getInteger()
{
    std::cout << "Enter an integer: ";
    int x{}; //define integer variable
    std::cin >> x; //get integer from console
    // std::cout << x << '\n'; // display integer
    return x;
}

// main.cpp; main module

#include <iostream>

//int input();
int getInteger();

int main()
{
    int x{ getInteger() };
    int y{ getInteger() };

    std::cout << x << " + " << y << " is " << x + y << '\n';
    return 0;
}
```

If I uncomment the cout line in input.cpp, it will accept keyboard input.  
With it commented, it locks up.

Any idea why?

👍 0    ➡ Reply



**Steven**

👤 Reply to [Neil](#) <sup>34</sup> ⌚ August 21, 2023 10:31 pm

You don't need the `//int input();` part as you're putting a forward declaration on the `int getInteger()` function.

From my understanding, you're just putting the function's name, and the IDE will automatically go to whatever .cpp file contains the function you're trying to call.

👍 0    ➡ Reply



**Neil**

👤 Reply to [Neil](#) <sup>34</sup> ⌚ August 8, 2023 8:38 pm

I should mention that I was using Visual Studio 2022  
-N

👍 0

↩ Reply



XX

💬 Reply to [Neil](#)<sup>35</sup> ⌚ August 11, 2023 9:41 am

It works fine for me

👍 0

↩ Reply

## Links

1. <https://www.learncpp.com/author/Alex/>
2. <https://www.learncpp.com/cpp-tutorial/forward-declarations/>
3. <https://www.learncpp.com/cpp-tutorial/naming-collisions-and-an-introduction-to-namespaces/>
4. [javascript:void\(0\)](#)
5. <https://www.learncpp.com/>
6. <https://www.learncpp.com/programs-with-multiple-code-files/>
7. <https://www.learncpp.com/cpp-tutorial/header-files/>
8. <https://gravatar.com/>
9. <https://www.learncpp.com/cpp-tutorial/programs-with-multiple-code-files/#comment-594921>
10. <https://www.learncpp.com/cpp-tutorial/programs-with-multiple-code-files/#comment-594280>
11. <https://www.learncpp.com/cpp-tutorial/programs-with-multiple-code-files/#comment-594139>
12. <https://www.learncpp.com/cpp-tutorial/programs-with-multiple-code-files/#comment-594214>
13. <https://www.learncpp.com/cpp-tutorial/programs-with-multiple-code-files/#comment-594495>
14. <https://www.learncpp.com/cpp-tutorial/programs-with-multiple-code-files/#comment-594559>
15. <https://www.learncpp.com/cpp-tutorial/programs-with-multiple-code-files/#comment-593937>
16. <https://www.learncpp.com/cpp-tutorial/programs-with-multiple-code-files/#comment-593533>
17. <https://www.learncpp.com/cpp-tutorial/programs-with-multiple-code-files/#comment-592993>
18. <https://www.learncpp.com/cpp-tutorial/programs-with-multiple-code-files/#comment-592679>
19. <https://www.learncpp.com/cpp-tutorial/programs-with-multiple-code-files/#comment-592566>
20. <https://www.learncpp.com/cpp-tutorial/programs-with-multiple-code-files/#comment-590533>
21. <https://www.learncpp.com/cpp-tutorial/programs-with-multiple-code-files/#comment-589649>
22. <https://www.learncpp.com/cpp-tutorial/programs-with-multiple-code-files/#comment-589536>
23. <https://www.learncpp.com/cpp-tutorial/programs-with-multiple-code-files/#comment-589043>
24. <https://www.learncpp.com/cpp-tutorial/programs-with-multiple-code-files/#comment-588608>
25. <https://www.learncpp.com/cpp-tutorial/programs-with-multiple-code-files/#comment-588225>
26. <https://www.learncpp.com/cpp-tutorial/programs-with-multiple-code-files/#comment-588256>

27. <https://www.learncpp.com/cpp-tutorial/programs-with-multiple-code-files/#comment-588259>
28. <https://www.learncpp.com/cpp-tutorial/programs-with-multiple-code-files/#comment-588283>
29. <https://www.learncpp.com/cpp-tutorial/programs-with-multiple-code-files/#comment-588375>
30. <https://www.learncpp.com/cpp-tutorial/programs-with-multiple-code-files/#comment-587260>
31. <https://www.learncpp.com/cpp-tutorial/programs-with-multiple-code-files/#comment-587325>
32. <https://www.learncpp.com/cpp-tutorial/programs-with-multiple-code-files/#comment-587040>
33. <https://www.learncpp.com/cpp-tutorial/programs-with-multiple-code-files/#comment-585917>
34. <https://www.learncpp.com/cpp-tutorial/programs-with-multiple-code-files/#comment-585259>
35. <https://www.learncpp.com/cpp-tutorial/programs-with-multiple-code-files/#comment-585303>